NON-PROVISIONAL APPLICATION FOR UNITED STATES PATENT

FOR

Power and/or Energy Optimized Compile/Execution

Inventor
Joel D. Munter
Murthi Nanja
Zhiguo Gao
Jin J. Xu

Attorney Docket No.: 111027-136084
IPN: P18606

Prepared by: Al AuYeung
Schwabe, Williamson & Wyatt, PC
Pacwest Center
1211 SW Fifth Ave., Suites 1600-1900
Portland, Oregon 97204

aauyeung@schwabe.com
503-796-2437

Express Mail Label No. ER084884008US

FIELD OF THE INVENTION

[0001] The present invention relates generally to the fields of data processing and data communication.

BACKGROUND OF THE INVENTION

[0002] Programming language compilers often include optimizers to improve the execution performance of the generated object code (also referred to as executable code or executables). The optimizations performed may include, for examples, removal of not executed codes ("dead code") to reduce the overall code size, loop unrolling for parallel execution to improve execution speed, and so forth.

[0003] In recent years, advances in microprocessor and related technologies have led to the development and availability of a wide range of wireless mobile devices, such as wireless mobile phones, personal digital assistants, and so forth. Concurrently, various software technologies, such as just-in-time compilation, and so forth, have been developed to facilitate cross platform application development.

[0004] The current state of just-in-time compilation has at least two disadvantages. First of all, it does not optimize for power level nor energy requirement of the wireless mobile devices. Secondly, the unconditional compilation of all received codes may be power and/or energy inefficient.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention will be described by way of exemplary embodiments, but not limitations, illustrated in the accompanying drawings in which like references denote similar elements, and in which:

[0006] **Figure 1** illustrates a block diagram view of a computing environment, in accordance with an embodiment of the present invention;

[0007] **Figure 2** illustrates a flow chart view of portions of the operational flow of the compiler of **Fig. 1** in accordance with one embodiment;

[0008] **Figure 3** illustrates a block diagram view of another computing environment, in accordance with another embodiment of the present invention; and

[0009] **Figure 4** illustrates a flow chart view of portions of the operational flow of the runtime manager of **Figure 3** in accordance with one embodiment.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0010] Illustrative embodiments of the present invention include but are not limited to a compiler with power and/or energy optimization, a complementary runtime manager, and a wireless mobile device having the compiler and/or the runtime manager.

[0011] Various aspects of the illustrative embodiments will be described using terms commonly employed by those skilled in the art to convey the substance of their work to others skilled in the art. However, it will be apparent to those skilled in the art that the present invention may be practiced with only some of the described aspects. For purposes of explanation, specific numbers, materials, and configurations are set forth in order to provide a thorough understanding of the illustrative embodiments. However, it will be apparent to one skilled in the art that the present invention may be practiced without the specific details. In other instances, well-known features are omitted or simplified in order not to obscure the illustrative embodiments.

[0012] Further, various operations will be described as multiple discrete operations, in turn, in a manner that is most helpful in understanding the present invention; however, the order of description should not be construed as to imply that these operations are necessarily order dependent. In particular, these operations need not be performed in the order of presentation.

[0013] The phrase "in one embodiment" is used repeatedly. The phrase generally does not refer to the same embodiment; however, it may. The terms "comprising", "having", and "including" are synonymous, unless the context dictates otherwise.

[0014] Referring now to **Figure 1**, wherein a block diagram view of an example computing environment, in accordance with one embodiment, is shown. As illustrated, example computing environment **100** includes processor **102** and memory **104** coupled to each other via bus **106** as shown. Memory **104** includes compiler **112**, which includes an optimizer equipped to optimize the object code **124** generated for an application module **122** to improve at least the power level requirement or the energy requirement for executing the object code **124** in a target execution environment. The target execution

environment may be computing environment **100** itself or another computing environment.

[0015] In various embodiments, application modules **122** may be provided to compiler **112** in a source form, e.g., Java, C#, or other languages of the like. In other embodiments, application modules **122** may be provided to compiler **112** in an intermediate form, e.g., a byte code form. In yet other embodiments, application modules **122** may be provided to compiler **112** in either form. Whether in source form or intermediate form, instructions of application modules **122** are "high level" instructions. In other words, they are non-native instructions of the target execution environment. The compilation generates object codes **124** comprised of native instructions of the target execution environment. The term "native instructions" as used herein refers to instructions of the instruction set supported by the processor of the target execution environment.

[0016] As described earlier, the compilation includes optimization for the power level requirement and/or energy requirement for executing generated object codes **124** in the target execution environment. For the purpose of this application, the term "power level requirement" refers to the highest power level required to execute a sequence of native instructions, which is typically the highest power level required by certain hardware circuitry to execute one or more of the native instructions of the code sequence. The power level required to execute a sequence of instructions affects the thermal dissipation of the target execution environment, as well as the life of the battery power source of the target execution environment (if it is powered by a battery power source). The term "energy requirement" refers to the amount of energy consumed to execute the sequence of native instructions in the target execution environment. Similarly, the energy required to execute a sequence of instructions also affects the life of the battery power source of the target execution environment (if it is powered by a battery power source).

[0017] In various embodiments, compiler **112** optimizes generated object code **124** to improve at least the power level required to execute the generated object code **124**. In other embodiments, compiler **112** optimizes generated object code **124** to improve at least the energy required to execute the generated object code **124**. In yet other embodiments, compiler **112** optimizes generated object code **124** to improve at least the

power level required as well as the energy required to execute the generated object code **124**.

[0018] In various embodiments, compiler **112** optimizes object code **124** in accordance with power and/or energy profiles **114**. More specifically, in various embodiments, power and/or energy profiles **114** comprise power and/or energy profiles of the native instructions of the target execution environment. For example, in one embodiment, if the native instructions of the target execution environment includes an ADD instruction, a SUBTRACT instruction, a MUTIPLY instruction, and so forth, power and/or energy profiles **114** include the power level required and/or the energy required to execute the ADD instruction, the SUBTRACT instruction, the MUTIPLY instruction, and so forth, respectively. Profiles **114** for a target execution environment may be empirically determined, and then provided to computing environment **100**.

[0019] Except for the teachings of the present invention, compiler **112** may be any one of a number of compilers known in the art or to be designed, including but are not limited to Java compiler, C# compiler, or a just-in-time (JIT) compiler. Application modules **122** may implement applications of any kind. Similarly, processor **102**, memory **104**, and bus **106** perform their conventional functions, and may be any one of a wide range of these elements known in the art or to be designed. The present invention may also be practiced in other computing environments with more or less hardware elements.

[0020] **FIGURE 2** illustrates a flow chart view of portions of the optimization operational flow of compiler **112**. As illustrated, upon given execution control, the optimizer portion of compiler **112** retrieves a sequence of instructions (also referred to as a code segment) for analysis, block **202**. The manner in which compiler **112** selects a code segment for analysis is language and application dependent. That is, for a language that supports certain language syntax for programming loops, on encounter of the start of a loop, compiler **112** may look for the end of the loop, and analyze the loop as a code segment. For a language that supports ADD and ACCUMULATE, on encounter of an ADD instruction, compiler **112** may look ahead for a predetermined number of instructions, and analyze them as a code segment to determine if some of the instructions

can be combined and replaced with other instructions that improve on one or more performance factors (e.g., size, execution speed, and so forth), and yet provide the same functional result(s).

[0021] Still referring to **Fig. 2**, on retrieval of a code segment, for the embodiment, the optimizer of compiler **112** analyzes the code segment for execution power level requirement, block **204**. As described earlier, in various embodiments, the analysis is performed in view of power profile **114**. In the course of analysis, the optimizer of compiler **112** determines if an alternative code segment with lower execution power level requirement is available, block **206**. If the determination is affirmative, the optimizer of compiler **112** replaces the code segment with the alternative code segment with lower execution power level requirement, block **208**.

[0022] On determination that no replacement code segment is available, block **206**, or on replacement, block **208**, for the embodiment, the optimizer of compiler **112** proceeds to analyze the code segment of execution energy requirement, block **210**. As described earlier, in various embodiments, the analysis is performed in view of energy profile **114**. In the course of analysis, the optimizer of compiler **112** determines if an alternative code segment with lower execution energy requirement is available, block **212**. If the determination is affirmative, the optimizer of compiler **112** replaces the code segment with the alternative code segment with lower execution energy requirement, block **214**.

[0023] On determination that no replacement code segment is available, block **212**, or on replacement, block **214**, the optimizer of compiler **112** proceeds to analyze the code segment for other optimization opportunities, block **216**.

[0024] While for ease of understanding, the sequence of optimizations has been orderly described with power level optimization first, followed by energy optimization, and other conventional optimizations, in alternate embodiments, other optimization orders may be practiced. For examples, energy optimization and/or other conventional optimization may be performed before power level optimization.

[0025] **Figure 3** illustrates a block diagram view of another example computing environment, in accordance with another embodiment of the present invention. As illustrated, similar to computing environment **100**, computing environment **300** includes

processor **302**, memory **304** coupled to each other via bus **306**. Further, computing environment **300** also includes communication interface **308** coupled to the earlier described elements as shown. Memory **304**, in addition to compiler **312**, power and/or energy profiles **314**, application module **322** and object codes **324** (which are corresponding to compiler **112**, power and/or energy profiles **114**, application module **122**, and object codes **124**), also includes interpreter **316** and runtime manager **318**, coupled to earlier described elements as shown.

[0026] Interpreter **316** is employed to interpretively execute application modules **322**. In various embodiments, interpreter **316** is equipped to interpretively execute application modules **322** in an intermediate form. In other embodiments, interpreter **316** is equipped to interpretively execute application modules **322** in source form. In yet other embodiments, interpreter **316** is equipped to interpretively execute application modules **322** in either source or an intermediate form.

[0027] In any event, to further optimize power usage and energy consumption in computing environment **300**, runtime manager **318** is used to immediately invoke interpreter **316** to interpretively execute a received application module **322** for at least an initial number of times, before invoking compiler **312** to compile application modules **322**, including optimizing generated code **324** to improve execution power level requirement and/or execution energy requirement in computing environment **300**.

[0028] The practice provides the advantage of increasing the likelihood in achieving a net saving in power level and/or energy, notwithstanding the power level and/or energy investments required to perform the compilation.

[0029] In various embodiments, computing environment **100** may further include sensors for sensing power level and/or energy consumption, and system services for reporting the sensing data collected. For these embodiments, runtime manager **318** may be further equipped to conditionally monitor the execution of object code **324** for the power level and/or energy requirements of the various native instructions, and update profiles **314** with the observed power level and/or energy requirement. The conditional monitoring and updating may be performed based at least in part on whether a hardware/software switch is set or not. Provision of the hardware/software switch, and

its setting may be provided and facilitated in any one of a number of known or to be designed manners.

[0030] In various embodiments, the initial number of times a received application module is to be executed before compilation is statically configured for runtime manager **318**. In other embodiments, runtime manager **318** may be further equipped to dynamically determine the initial number of times a received application is to be executed before compiling the application. Runtime manager **318** may be equipped to make the determination based at least in part on the power level required and/or energy required to perform an "average" compile. Typically (though not necessarily), the higher power level or the more energy required to perform an "average" compile, the more times a received application module will be interpretively executed before being compiled. Runtime manager **318** may also make the decision further based on the size of the received application module **322**. Similarly (though not necessarily), the larger the received application module (implying more energy is required to perform a compile), the more times a received application module will be interpretively executed before being compiled.

[0031] In some of these embodiments, runtime manager **318** may be further equipped to conditionally monitor the compilation of application modules **322** for the power level and/or energy required to perform an "average" compile, and update the number of initial executions to be performed before compilation accordingly, based at least in part on the results of the monitoring. Similarly, the conditional monitoring and updating may be performed based at least in part on whether a hardware/software switch is set or not. Again, provision of the hardware/software switch, and its setting may be provided and facilitated in any one of a number of known or to be designed manners.

[0032] In various embodiments, communication interface **308** is a wireless communication interface, and computing environment **300** is a wireless mobile device, such as a wireless mobile phone or a wireless personal digital assistant.

[0033] **Figure 4** illustrates a flow chart view of portions of the operational flow of runtime manager **318**, in accordance with one embodiment. The embodiment assumes the computing environment is equipped with the proper sensor hardware and sensed data

reporting services. As illustrated, for the embodiment, on receipt of an application module **322**, runtime manager **318** first determines a number of times (N) interpreter **316** should be used to interpretively execute the received application module **322** before compilation, block **402**. On determination (and assuming N is non-zero), runtime manger **318** invokes interpreter **316** to interpretively execute the received application module **322** for up to N times, block **404**.

[0034] During this period, runtime manager **318** monitors the execution. If indeed the received application module **322** has been executed up to N times, runtime manager **318** invokes compiler **312** to compile the received application module **322** into object code **324**, including optimizing object code **324** to improve at least execution power level requirement or execution energy requirement, block **406**. In various embodiments, as described earlier, on invocation of compiler **312**, runtime manager **318** may monitor the compilation for the power level and/or energy required to perform an "average" compile, and update the profile accordingly, if a hardware/software switch is set.

[0035] Thereafter, runtime manager **318** allows object code **324** to be executed for as long as it is needed, block **408**. For the embodiment, at the same time, runtime manager **318** conditionally monitors the execution of the object code **318** for at least part of the time, and updates the profiles accordingly, blocks **410-412**, if a hardware/software switch is set.

[0036] Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that a wide variety of alternate and/or equivalent implementations may be substituted for the specific embodiments shown and described, without departing from the scope of the present invention. This application is intended to cover any adaptations or variations of the embodiments discussed herein. Therefore, it is manifestly intended that this invention be limited only by the claims and the equivalents thereof.